VisIVOServer 1.0

(subversion 1.0.0.4)

User Guide

Introduction

Authors: Becciani U., Caniglia G., Costa A., Gheller C., Krokos M., Massimino P.

VisIVO Server is a suite of software tools for creating customized views of 3D renderings from astrophysical data tables. These tools are founded on the **VisIVO Desktop** functionality (visivo.oact.inaf.it) and support the most popular Linux based platforms (e.g. www.ubuntu.com). Their defining characteristic is that no fixed limits are prescribed regarding the dimensionality of data tables input for processing, thus supporting very large scale datasets.

VisIVO Server websites are currently hosted by the University of Portsmouth, UK (visivo.port.ac.uk), the INAF Astrophysical Observatory of Catania, Italy (visivo.oact.inaf.it) and in the near future by CINECA, Italy (visivo.cineca.it). These web sites offer data management functionality for registered users; datasets can be uploaded for temporary storage and processing for a period of up to two months. The sites can also be utilized through anonymous access in which case datasets can be uploaded and stored for a maximum of four days; to maximize available resources a limited dimensionality is only supported.

Assuming that datasets are uploaded, users are typically presented with tree-like structures (for easy data navigation) containing pointers to **files**, **tables**, **volumes** as well as **visuals**.

Files point to single, or possibly several (for distributed datasets), astrophysical data tables;
Tables are highly-efficient internal VisIVO Server data representations; they are typically produced from importing datasets uploaded by users using VisIVO Importer (see below);
Volumes are internal VisIVO Server data representations; they are produced either from direct importing of user datasets or by performing operations on already existing tables;
Visuals are collections of highly-customized, user-produced views of 3D renderings of volumes.

VisIVO Server consists of three core components: **VisIVO Importer**, **VisIVO Filter** and **VisIVO Viewer** respectively. Their functionality and usage is described in the following sections.

To create customized views of 3D renderings from astrophysical data tables, a two-stage process is employed. First, VisIVO Importer is utilized to convert user datasets into **VisIVO Binary Tables** (VBTs). Then, VisIVO Viewer is invoked to display customized views of 3D renderings. As an example, consider displaying views from only three columns of an astrophysical data table supplied in ascii form, say col_1, col_2 and col_3, by using the commands

```
VisIVOImporter --fformat ascii UserDataSet.txt

VisIVOViewer -x col_1 -y col_2 -z col_3 --scale --glyphs pixel VBT.bin
```

VisIVOServer is distributed with GPL V.2 License for NON COMMERCIAL use. VisIVOServer is hosted by sourceforge https://sourceforge.net/projects/visivoserver/ and its source code is downloadable via svn:

```
svn co https://visivoserver.svn.sourceforge.net/svnroot/visivoserver visivoserver
```

Disclaimer: user data integrity is never warranted.

VisIVO BINARY TABLE

A VisIVO Binary Table (VBT) is a highly-efficient data representation used by VisIVO Server internally. A VBT is realized through a header file (extension **.bin.head**) containing all necessary metadata, and a raw data file (extension **.bin**) storing actual data values. For example, the header may contain information regarding the overall number of fields and number of points for each field (for point datasets) or the number of cells and relevant mesh sizes (for volume datasets). The raw data file is typically a sequence of values, e.g. all X followed by all Y values.

Header

The header file contains the following fields:

```
float | double
n1
n2 [ GeoX GeoY GeoZ DX DY DZ ]
little | big
X
Y
Z
Vx
Vy
Vz
```

- **float | double** is the data type of the storage variables used;

- **n1** denotes the number of columns (fields) in the VBT;
- **n2** denotes the number of rows in the VBT;
- ***Geo*X GeoY GeoZ DX DY DZ** are employed only if the VBT represents volumetric datasets. In that case GeoX, GeoY and GeoZ represent the mesh geometry, while DX, DY and DZ represent the x, y and z size of volumetric cells.
- **little | big** denotes the endianism employed in the VBT. After this field there exist n1 rows that indicate the VBT columns as positions (X, Y, Z) and velocities (Vx, Vy, Vz).

Raw

The binary file is simply a sequence of *n1*n2* values. In the example shown in section 2.1 all X values, then all Y values and so on.

Note:

- **n1** represents the number of columns (fields) in the VBT (e.g. 6);
- **n2** represents the number of elements of each field in the VBT (e.g. 262144);
- **GeoX GeoY GeoZ** represent the number of the volumetric cells in each dimension of the mesh size (used only for Volumes) (e.g. 64 64 64);
- **DX DY DZ** represent the size of each cell (used only for Volumes) (e.g. 1.0 1.0 1.0)

VisIVOServer 1.0

(subversion 1.0.0.4)

VisIVO Importer

VisIVO Importer converts user-supplied datasets into a VBT. VBTs are used by VisIVO Filters for data processing and the VisIVO Viewer for display. The conversion is independent of data dimensionality. The VisIVO Importer command flags are explained in detail in the rest of this section.

General VisIVO Importer Syntax

VisIVOImporter --flag1 --flag2 ... --flagn FileName

--fformat specifies the format of input datasets;
--out [name] (optional) OutputFileName. The path specified with this flag is the VisIVO Importer output directory. If no path is specified the VisIVO Importer output directory is the current directory, and the default name for **OutputFileName** is **VisIVOServerBinary.bin**;

--**volume** (optional) is used to create volumes;

--**compx** [double] --**compy** [double] --**compz** [double] (optional) is used for entering geometry for volumetric datasets; if data size fits in a cubic mesh, values are computed automatically;

--**sizex** [double] --**sizey** [double] --**sizez** [double] (optional) is used for specifying volumetric cell dimensions; the default values are 1.0, 1.0 and 1.0 respectively;

--**userpwd** (optional) is used to prescribe a username and password for accessing remote files;
--**binaryheader** (optional) is used to specify the file name of the header of VBTs; this flag is ignored in case of other formats;

--**bigendian** (optional) is used only for big endian Gadget and FLY input files, otherwise by default this flag is set to little endian;

--**double** (optional) is used only for the double data type of FLY input files, otherwise by default this flag is set to float;

--**npoints** (optional) is used only for FLY input files to specify the number of data points.

The FileName is the local (or remote) file containing the data to be converted into a VBT. If FileName starts with *http:// , ftp:// or sftp://* the remote file is downloaded automatically. However if the --userpwd option is specified, the prescribed username and password are employed for remote access.
NOTE: the server must have libcurl with ssl support to enable the sftp functionality:

Example:

VisIVOImporter --fformat ascii –-out mytable sftp://machinename.domain/home/user/asciifile.txt.

The sftp syntax requires the remote directory where data are located.
All downloaded files are copied temporarily into the directory given in --out option and are deleted automatically at the end of the import process. Under the current file directory, the file *DownLoadedFilename_VisIVO_List* contains information on all download operations. If the fformat option is *binary* VisIVOImporter will attempt to download two remote files, a binary table (given as remote filename) and its associated header file (same name +".head" extension).

VisIVO Importer

ASCII and CSV FORMAT

CSVASCII files are expected to be in tabular form. An ASCII file may contain values for N variables organized in columns. The columns are typically separated by whitespace characters, e.g. spaces or tabs. The first row of an ASCII file lists the N variables names explicitly. As an example, the command below produces *NewTable.bin*, *NewTable.bin.head* from *ASCIIUserFileName.txt*.

Syntax Example

```
VisIVOImporter --fformat ascii --out /home/user/data/NewTable ASCIIUserFileName.txt
```

CSV is a delimited data format that has fields/columns separated by the comma character and records/rows separated by newlines. Fields that contain a special character (such as comma, newline, or double quote) must be enclosed in double quotes. However, if a line contains a single entry that happens to be the empty string, it may be enclosed in double quotes. If a field's value is a double quote character, this is dealt with by placing another double quote character next to it. The CSV file format does not require a specific character encoding, byte order, or line terminator format. As an example, the command below produces the files *NewTable.bin* and *NewTable.bin.head* from the user-supplied CSV file *CSVUserFileName.txt*.

Syntax Example

```
VisIVOImporter --fformat csv --out /home/user/data/NewTable CSVUserFileName.txt
```

VisIVO Importer

BINARY FORMAT

The binary format is supposed to be the Internal Binary Table

Syntax Example:

VisIVOImporter --fformat binary --out /home/user/data/NewTable BinaryUserFilename.bin

Assuming that a VBT is to be processed, the previous command produces *NewTable.bin* and *NewTable.bin.head* from *VBTUserFileName.bin*. Note that the files VBTUserFileName.bin and VBTUserFileName.bin.head must be (either local or remote) existing files. Also, if the –-binaryheader option is prescribed, a header file with the specified filename must exist. In case this header file resides remotely, a complete copy is created within the VisIVO Server output directory.

Note

This command is useful for changing the endianism of a binary table. An input big endian table is transformed in a little endian table if system where VisIVOImporter is running is a little endian system and viceversa.

VisIVO Importer

VOTable FORMAT

The VOTable format is an XML standard for the interchange of data represented as a set of tables. In this context, a table is an unordered set of rows, each having a uniform format, as specified in the table metadata information. Each row in a table is a sequence of table cells, and each of these contain either a primitive data type or an array of such primitives. It can also contain a link to an external file, that the XML part describes. No VOTables with binary values are supported in VisIVO.

The file sizes that can be processed by the VisIVO Server are only limited by the underlying parsing libraries. As an example, the command below produces *NewTable.bin*, *NewTable.bin.head* from *VOTableUserFileName.xml*.

The current version of VisIVO Server can read files with no more than 200MB of data.

Syntax Example:

VisIVOImporter --fformat votable --out /home/user/dataNewTable VOTableUserFilename.xml

VisIVO Importer

FLY FORMAT

FLY is code that uses the tree N-body method, for three-dimensional self-gravitating collisionless systems evolution. FLY is a fully parallel code based on the tree Barnes-Hut algorithm; periodical boundary conditions are implemented by means of the Ewald summation technique.
FLY is based on the one-side communication paradigm for sharing data among processors, accessing remotely private data without synchronism.
The FLY output format is a binary sequence of values of n data points as follows: X1,Y1,Z1, X2,Y2,Z2 ,.... Xn,Yn,Zn, Vx1,Vy1,Vz1, Vx2,Vy2,Vz2, ... Vxn,Vyn,Vzn. As an example, the command below produces *NewTable.bin* and *NewTable.bin.head* from FLYUserFile which is a FLY file using double data types, containing a total of 2000000 data points.

Syntax Example

```
VisIVOImporter --fformat fly --out /home/user/data/NewTable --double --npoints
2000000 FLYUserFile
```

The FLY format also allows the download of elements given in a descriptor file (.desc extension):

```
flyDesc     (type of the Fly descriptor)
2m_test     (ID)
double      (data type)
time        (ID for snapshot sequence)
2097152     (number of points on each snapshot)
50 50 50    (box dimension in the proper unit)
l           (“l” or “b” for data endianism)
0.0 out_scdm_0.0000 (time tag and  snapshot filename sequence)
1.0 out_scdm_1.0000
2.0 out_scdm_2.0000
```

In this case the flags --npoints and --double must no be given as values are read automatically from the descriptor file. Each listed file (out_scdm_# in this case) produces a VBT.

Syntax Example

```
VisIVOImporter --fformat fly --out /home/user/data/NewTable FLYUserFile.desc
```

Note that names of output files will be determined by using the --out parameter concatenated first by a "_" and then by the listed filename. As an example, if --out /tmp/pippo is prescribed, output filenames will be /tmp/pippo_out_scdm_0.000.bin (and one with the extension.bin.head). On the other hand, if --out option is not prescribed,

output filenames will be ./VisIVOServerBinaryout_scdm_0.000.bin (and one with extension .bin.head) .

VisIVO Importer

FITS Table FORMAT

The definition of FITS is a codification into a formal standard, by the NASA/Science Office of Standards and Technology (NOST), of the FITS rules (http://fits.gsfc.nasa.gov) endorsed by the IAU. FITS supports tabular data with named columns and multidimensional rows. Both binary and ASCII FITS table versions have been specified. The data in a column of a FITS table can be in a different format from the data in other columns. Together with the ability to string multiple header/data blocks together, by using FITS files it is possible to represent entire relational databases. As an example, the command below produces *NewTable.bin*, *NewTable.bin.head* from FITSTableUserFile.

Syntax Example

```
VisIVOImporter --fformat fitstable --out /home/user/data/NewTable
FITSTableUserFile
```

VisIVO Importer

FITS Image FORMAT

The Definition of FITS is a codification into a formal standard, by the NASA/Science Office of Standards and Technology (NOST), of the FITS rules endorsed by the IAU. FITS image headers can contain information about one or more scientific coordinate systems that are overlain on the image itself. Images contain an implicit Cartesian coordinate system that describes the location of each pixel in the image, but scientific uses generally require working in 'world' coordinates, for example the celestial coordinate system.

Under Development

VisIVO Importer

GADGET FORMAT

GADGET is freely-available code for cosmological N-body/SPH simulations on massively parallel computers with distributed memory. GADGET uses an explicit communication model that is implemented with the standardized MPI communication interface. The code can be run on essentially all supercomputer systems presently in use, including clusters of workstations or individual PCs.

VisIVO Importer will produce a VBT for each species in the gadget file supplied. As an example, the command below will produce the files *NewTableHALO.bin*, *NewTableHALO.bin.head*, and *NewTableGAS.bin*, *NewTableGAS.bin.head* from the GADGET file GadgetUserFile if we assume that it contains only two species, namely halo and gas particles.

Syntax Example

```
VisIVOImporter --fformat gadget --out /home/user/data/NewTable GadgetUserFile
```

VisIVO Importer

HDF5 FORMAT

Hierarchical Data Format, is a library and multi-object file format for the transfer of graphical and numerical data between computers. It was created by the NCSA, but is currently maintained by The HDF Group. The freely available HDF distribution consists of the library, command-line utilities, test suite source, Java interface, and the Java-based HDF Viewer (HDFView). HDF supports several different data models, including multidimensional arrays, raster images, and tables. Each defines a specific aggregate data type and provides an API for reading, writing, and organizing the data and metadata. New data models can be added by the HDF developers or users.

Under Development

VisIVO Importer

RAW Binary FORMAT

Raw files are simply a binary dump of the memory for data points. The content of the Raw Binary data points file is a sequence of x,y and z coordinate for each point, then a sequence of fields, one scalar for each data point.

General binary file structure:

X1, Y1, Z1
X2, Y2, Z2
.....
Xn, Yn, Zn

Field0_1
Field0_2
.....
Field0_n

Field1_1
Field1_2
……..
Field1_n
……..

Fieldm_1
Fieldm_2
……..
Fieldm_n

VisIVOImporter reads a descriptor file. More than one raw data file name can be described. The descriptor file has the following structure:

- rawPointDesc
- Variable name
- Variable type
- Time Variable (at the moment not used, but needed in the descriptor file)
- Number of particles
- The size of the box
- Endianism type (b=big endian or l=little endian)
- List of Ids of the data files (a number representing the order or the time) and names of the data files

Example of Descriptor file

rawPointsDesc
dark
Float
time
130000
50 50 50
b
0.0 16ml_096
0.5 16ml_104
1.0 16ml_112

All the files listed in the descriptor file must be given. Each file will be converted and an internal binary table will be created for each listed file
Output files will have the same name of --out parameter +listedfilename+".bin" and ".bin.head". If a filename start with *http://* or *ftp://* the remote file is downloaded. If the --userpwd option is given the username and password are used for remote access to the file. Downloaded files are temporarily copied in the same directory given in --out option and cleaned at the end of the import phase. The file *downLoadedFilename_VisIVO_list* in the current directory contains information on the download operations.

Syntax Example

VisIVOImporter --fformat rawpoints rawpointsUserFile.desc

VisIVO Importer

RAW Grid FORMAT

Raw files are simply a binary dump of the memory. For volume data, only one quantity is expected to be stored in each file. The content of a volume file is a sequence of values: one value for each mesh point.

VisIVOImporter reads a descriptor file. More than one raw data file name can be described. The descriptor file has the following structure:

- rawGridDesc
- Variable Name
- Number of spatial dimensions
- Variable type
- Number of cells in the first dimension
- Number of cells in the second dimension
- Number of cells in the third dimension
- Time variable (in the present release not used, but required in the descriptor file)
- Endianism type (b=big endian or l=little endian)
- List of Ids of the data files (a number representing the order or the time) and names of the data files

Example of Descriptor file:

rawGridsDesc
density
3
Float
64
64
64
time
l
0.0 JET8Xhj.f064.dat.pff
0.7 JET8Xeh.f064.dat.pff
2.1 JET8Tat.f064.dat.pff

All the files listed in the descriptor file must be given. Each file will be converted and an internal binary table will be created for each listed file.
Output files will have the same name of --out parameter +listedfilename+".bin" and ".bin.head". If a filename start with *http://* or *ftp://* the remote file is downloaded. If the --userpwd option is given the username and password are used for remote access to the file.

Downloaded files are temporarily copied in the same directory given in --out option and cleaned at the end of the import phase. The file *downLoadedFilename_VisIVO_list* in the current directory contains information on the download operations.

Syntax Example:

VisIVOImporter --fformat rawgrids rawpointsUserFile.desc

VisIVO Importer

XML DOC FORMAT

VisIVOImporter can read an XML file as reported below and can download local or remote files and convert them into one or more output binary internal data format files. This document is a VOTable that must contain all the key information to download data. Each row in the TABLEDA <TR> ... </TR> will be a field of the output file. All the rows with the same species identifier are included in the same output file.

If the Arraysize field has only one value a data point field is assumed. If Arraysize has three values a volume field is assumed.
Downloaded files have the same name of the remote file +"_VisIVOImporter_"+*progressive number* . The produced files (binary internal data format) have the following filenames: --out file root +"_"+*Species*+".bin" and ".bin.head"

If a filename starts with *http://* or *ftp://* the remote file is downloaded. If the --userpwd option is given the username and password are used for remote access to the file.
Downloaded files are temporarily copied in the same directory given in --out option and cleaned at the end of the import phase. The file *downLoadedFilename_VisIVO_list* in the current directory contains information on the download operations.

FIELD description

Format

The XML file could contain the description of each field of the local or remote file. If the XML file does not contain information on the fields it can refer to a specified file format. In this case the Format key can contain one of the following values:

ascii csv fitstable votable gadget hdf5

If a different key value is given, the remote file will be considered a binary file and all the below field must be given.

Species

Name of species. Each row with the same Species Id will be a field of the same out table file.

Field

Column Name reported in the output table

Unit

Unit of the specified file.

Offset

Byte to be skipped for the first value of the field.

Stride

Byte to be skipped between two consecutive values of the same field

Endianism

LittleEndian or BigEndian

Rank

Rank of the field. The Importer produce one column for each rank

Arraysize

Number of elements in the specified field. It must contain one value if the field is a data point, three values if the field is a volume. In the latter case each value represents the 3D mesh size.

Type

Internal data representation. Allowed values: float, double, long double, int, long int and long long int

Precision

Number of Bytes used for the above Type. It must match the same representation of the system where VisIVOImporter is running.

Syntax Example:

VisIVOImporter --fformat xml userFile.xml

XML Example file

```
<?xml version="1.0" encoding="UTF-8"?>
<VOTABLE xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://www.ivoa.net/xml/VOTable/v1.1"
  xsi:schemaLocation="http://www.ivoa.net/xml/VOTable/v1.1
http://www.ivoa.net/xml/VOTable/v1.1">
 <DESCRIPTION>
        This is a first TVO xml (VOTable) prototype for Theoretical data
 </DESCRIPTION>


<INFO ID="Ref" name="-ref" value="VOTx1"/>

<RESOURCE ID="ITVO_Catania_Server">

<DESCRIPTION>Theoretical data from Catania Cosmological Simulations</DESCRIPTION>

<LINK href="http://itvo.oact.inaf.it/"> </LINK>
<TABLE name="MyQueryResults" ID="Result">
  <PARAM name="Simulation0003_12" ID="mySimId"  datatype="char" arraysize="*"
value="Simulation0003_12">
  </PARAM>
  <FIELD name="Time" UNIT="Redshift" datatype="float" ucd="" precision="4" utype=""/>
  <FIELD name="Format" datatype="char"/>
  <FIELD name="Species"  datatype="char"/>
  <FIELD name="Field"  datatype="char"/>
  <FIELD name="Unit"  datatype="char"/>
  <FIELD name="Offset"  datatype="long"/>
  <FIELD name="Stride" datatype="long"/>
  <FIELD name="Endianism" datatype="char"/>
  <FIELD name="Rank" datatype="int"/>
  <FIELD name="Arraysize" datatype="char"/>
  <FIELD name="Type" datatype="char"/>
  <FIELD name="Precision" datatype="int"/>
  <FIELD name="href" datatype="char"/>
  <DATA>
    <TABLEDATA>

<TR><TD>0.0</TD><TD>CUSTOM</TD><TD>DM</TD><TD>xcoordinate</TD><TD>Mpc
</TD><TD>0</TD><TD>8</TD><TD>LittleEndian</TD><TD>1</TD><TD>10000</TD>
<TD>float</TD><TD>4</TD><TD>ftp://astrct.oact.inaf.it/pub/becciani/Binary/out_2ml.bin
</TD></TR>
```

<TR><TD>0.0</TD><TD>CUSTOM</TD><TD>DM</TD><TD>ycoordinate</TD><TD>Mpc
</TD><TD>4</TD><TD>8</TD><TD>LittleEndian</TD><TD>1</TD><TD>10000</TD>
<TD>float</TD><TD>4</TD><TD>ftp://astrct.oact.inaf.it/pub/becciani/Binary/out_2ml.bin
</TD></TR>

<TR><TD>0.0</TD><TD>CUSTOM</TD><TD>DM</TD><TD>zcoordinate</TD><TD>Mpc
</TD><TD>8</TD><TD>8</TD><TD>LittleEndian</TD><TD>1</TD><TD>10000</TD>
<TD>float</TD><TD>4</TD><TD>ftp://astrct.oact.inaf.it/pub/becciani/Binary/out_2ml.bin
</TD></TR>

<TR><TD>0.0</TD><TD>CUSTOM</TD><TD>DM</TD><TD>mass</TD><TD>SolarMass
es</TD><TD>120000</TD><TD>0</TD><TD>LittleEndian</TD><TD>1</TD><TD>1000
0</TD><TD>float</TD><TD>4</TD><TD>/home/user/ Binary/out_2ml.bin</TD></TR>
</TABLEDATA>
</DATA>
</TABLE>
</RESOURCE>
</VOTABLE>

VisIVOServer 1.0

(subversion 1.0.0.4)

VisIVO Filters

VisIVO Filters are programs that convert data from an original data table (internal data format table) into a new one or can create a Volume from a table. Filters are directly applied by the user using VisIVO Server Website: the web procedure directly calls the filters when applicable.

GENERAL SYNTAX:

VisIVOFilters --help
produces a general help

VisIVOFilters --op operation --help
produces the help of a specific operation

VisIVOFilters --op operation <options> [--file] InputFile
runs the operation

Where requested the InputFile is a valid table (Binary Internal data format). In this case the *InputFile.bin* and *InputFile.bin.head* must exist.

The following filter operations are available

Append Tables
creates e new table appending data from a list of existing tables

Coarse Volume
produces a coarsed subvolume with plane extraction from the original volume

Decimator
creates a subtable as a regular subsample from the input table.

Extract Subregion
creates a new table from an input table of a sub-box or of a sphere.

Extract Subvolume
produces a table which represents a subvolume from the original volume.

Interpolate
creates new tables from two existing data tables with a linear interpolation.

Math. Operations
creates a new field in a data table as the result of a mathematical operation between the existing fields.

Merge Tables
creates a new table from two or more existing data tables.

Point Distribute
creates a volume using a field distribution (CIC algorithm) on a regular mesh.

Point Property
assigns a property to each data point on the table

Randomizer
creates a random subset from the original data table.

Select Fields
creates a new table using one or more fields of a data table.

Select Rows
creates a new table using limits on one or more fields of a data table.

Show Table
produces an ASCII table of a selected field of the first number of rows (or of all rows).

Statistic
creates and plots an histogram of a scalar field of a data table.

Visual
creates a randomized new table from an input table listing tables and columns with equal data size, to be used with VisIVOViewer.

VisIVO Filters

Append Tables

This operation creates e new table appending data from a list of existing tables. Append Filter can append up to 100 tables with the same number of Columns

Usage:

VisIVOFilters --op append [--out filename_out.bin] [--help] [--file] table_list.txt

Example:

VisIVOFilters --op append --out out_table.bin --file tab_list.txt

tab_list.txt is a file that contains a list of valid table names. The ".bin" extension is automatically added if the listed filename does not contain it.

/home/user/tab1.bin
/home/user/tab2.bin
...
/home/user/otherDirectory/tabN.bin

The filter produces a new table (*out_table.bin* and *out_table.bin.head*). The column names are copied from the first table. An error is given if tables contain different numbers of columns.

Note:
--out Name of the new table. Default name is given.
--file An ascii file with a valid list of tables

VisIVO Filters

Coarse Volume

This operation produces a coarse subvolume with plane extraction from the original volume.

Usage:

VisIVOFilters --op coarsevolume [--perc percentage] [--newres x_res y_res z_res] [--field column_names] [--out filename_out.bin] [--help] [--file] inputFile.bin

Example:

VisIVOFilters --op coarsevolume --perc 10.0 --field Mass Temperature --out subvolume.bin --file inputFile.bin

produces a new table (*subvolume.bin* and *subvolume.bin.head*). It is a volume of 10 percent of the original volume. If the original volume is a mesh of 320x320x640 the output volume will be 32x32x64 and only the listed fields Mass and Temperature will be considered. Planes are extracted from the original volume as a regular distribution from the original mesh.

Note:
--perc A percentage (from 0.0 to 100.0) subvolume will be produced. Default value produces a sub volume that could be directly uploaded and visualized with VisIVOViewer and VisIVODesktop applications.
--field List of columns contained in the original file. Default: all columns will be extracted.
--newres A subvolume with new resolution will be produced. No default is given. This parameter is ignored if --perc option is given
--out Name of the new table. Default name is given.
--file Input table filename.

VisIVO Filters

Decimator

This operation creates a subtable as a regular subsample from the input table.

Usage:

VisIVOFilters --op decimator --skip step [--out filename_out.bin] [--help] [--file] inputFile.bin

Example:

VisIVOFilters --op decimator --skip 9 --out inputFile_samp.bin --file inputFile.bin

produces a new table (*filename_out.bin* and *filename_out.bin.head*) having a 10 percent size of the original inputFile. Values are extracted in a regular sequence, skipping step element every time. The skip value is an integer number > 1 and represents the number of skipped values. In the above example only one element every 10 elements will be reported in the output file. The output table must fit the available RAM.

Note:
--skip Integer representing the skipped values
--out Name of the new table. Default name is given.
--file Input table filename.

VisIVO Filters

Extract Subregion

This operation creates a new table from an input table of a sub-box or of a sphere. Operation not allowed on volumes.

Usage:

VisIVOFilters --op extraction --geometry geometry_file [--out filename_out.bin] [--help] [--file] inputFile.bin

Example:

VisIVOFilters --op extraction --geometry geometry.txt --out pos_extracted.bin --file inputFile.bin

The geometry.txt file must have four rows and two columns. The first three rows have a valid column name and a value that indicates the extraction coordinates:

X 20.0
Y 20.0
Z 20.0
CORNER 10.0

The command produces a new table (*pos_extracted.bin* and *pos_extracted.bin.head*) that contain all data points of the inputFile table (all columns will be reported) that are included in the box having the lower corner at X=20.0 Y=20.0 and Z=20.0 and size=10.0

Note:

--geometry is a file name that contains three valid column names and a value for each column. The fourth field means the extraction mode and the subvolume size:

RADIUS: a sphere centered in the given values will be extracted
CORNER: a rectangular region having the lower corner at the given values will be extracted
BOX: a rectangular region centered in the given values will be extracted.

Examples:

X 25.0
Y 25.0
Z 25.0
RADIUS 5.0

or

X 25.0
Y 25.0
Z 25.0
BOX 5.0

or

X 0.0
Y 0.0
Z 0.0
CORNER 10.0

--out Name of the new table. Default name is given.
--file Input table filename.

VisIVO Filters

Extract Subvolume

This operation extract a table which is a subvolume from the original volume

Usage:

VisIVOFilters --op extractsubvolume --startingcell X Y Z --resolution x_res y_res z_res [--field column_names] [--out filename_out.bin] [--help] [--file] inputFile.bin

Example:

VisIVOFilters --op extractsubvolume --startingcell 8 8 8 --field Mass Temperature --resolution 16 16 16 --out mysubvolume.bin --file inputFile.bin

produces a new table volume (*mysubvolume.bin* and *mysubvolume.bin.head*) that is a subvolume of resolution 16x16x16 from the original volume and starting from the cell (8,8,8) of the original mesh. Only Mass and Temperature fields will be reported in the new table.

Note:
--startingcell X Y Z number of the first cell to be extracted: 0 0 0 is the first cell of the original grid
--resolution Grid size (3D) of the new subgrid
--field Valid columns name list to be reported in the new table.
--out Name of the new table. Default name is given.
--file Input table filename.

VisIVO Filters

Interpolate

This operation creates new tables from two existing data tables (mainly used to produce intermediate frames of a dynamical evolution).

Usage:

VisIVOFilters --op interpolate [--list columns_name] [--numbin numberbin] [--periodic] [--interval from to] [--out filename_out] [--help] --infiles file_start.bin file_end.bin

Example:

VisIVOFilters --op interpolate --list X Y Z --numbin 20 --periodic --out mysequence --infiles start.bin end.bin

This command produces new tables (mysequence_0.bin, mysequence_1.bin .... mysequence_19.bin and the header files) having only the three fields X, Y and Z. This Filter creates a file in the same directory of the -–out file VSInterpolatefilelist_time.txt that lists all names on created files, where time is the current time.

Limits:
The infiles tables must have the listed columns in the --list option in the same corresponding order. If Y is the second column of the start.bin file, the end.bin file must contain Y as the second column. The input tables must have the same number of rows and the interpolated elements are considered in the same order. No index is currently supported.

Note:
--list a valid list of columns names that must exist on both input tables. Default: all columns in infile files are considered.
--numbin is the number of bins between the file_start.bin and the file_end.bin input files or the interval given in the --interval option. The default value is 10. The number of created tables is equal to numberbin-1.

--periodic applies a periodical boundary condition
--interval. VisIVO assumes a distance of 1.0 between the two input frames: file_start.bin and file_end.bin. This option produces the intermediate frames (tables) in a subinterval between the two input frames. The value 0.5 is the medium point of the interval. If the from value is lower than 0.0 it is considered 0.0. If the to value is lower than 1.0 it is considered 1.0. If the from value is equal to to value the operation is not performed. Default value from=0.0 to =1.0
--out is the root name of the new tables. The default name is given. The new name is given by the filename_out#.bin where # is the number of created tables.
--infiles contains the names of the input tables of the interpolation process.

VisIVO Filters

Mathematical Operations

It is based on Function parser for C++ v2.83 by Warp (http://iki.fi/warp/FunctionParser/) with some minor modifications. Create a new field in a data table as the result of a mathematical operation between the existing fields

Usage:

VisIVOFilters --op mathop --expression math_expression.txt [--append] [--outcol col_name] [--out filename_out.bin] [--help] [--file] inputFile.bin

Example:

VisIVOFilters --op mathop --expression my_expression.txt --outcol SqrMod --out SqrModTable.bin --file inputFile.bin

my_expression.txt is a file that contains only one row with a mathematical expression:

*sqrt(VelX*VelX+VelY*VelY+VelZ*VelZ)*

being *VelX VelY* and *VelZ* columns name listed in *inputFile.bin.head*

The command produces a new table (*SqrModTable.bin* and *SqrModTable.bin.head*) with only one column called SqrMod having the same length of the columns of the inputFile.bin table

Note:
A new field is produced.
--expression A file with only one row having any valid mathematical expression with Valid Column names
--append No new table will be created. The original table will have the new field. Default options: a new table with only the new field is produced.
--outcol. Column name of the new field
--out Name of the new table. Default name is given. Ignored if --append is specified.
--file Input table filename

As reported in the Function parser library, function string is very similar to the C-syntax. Arithmetic float expressions can be created from float literals, variables or functions using the following operators in this order of precedence:

()	expressions in parentheses first
A unit	a unit multiplier (if one has been added)
A^B	exponentiation (A raised to the power B)
-A	unary minus
!A	unary logical not (result is 1 if int(A) is 0, else 0)
A*B A/B A%B	multiplication, division and modulo
A+B A-B	addition and subtraction
A=B A!=B A<B A<=B A>B A>=B	comparison between A and B (result is either 0 or 1)
A&B	result is 1 if int(A) and int(B) differ from 0, else 0
A|B	result is 1 if int(A) or int(B) differ from 0, else 0

Since the unary minus has higher precedence than any other operator, for example the following expression is valid: x*-y

The comparison operators use an epsilon value, so expressions which may differ in very least-significant digits should work correctly. For example, "0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1 = 1" should always return 1, and the same comparison done with ">" or "<" should always return 0. Without epsilon this comparison probably returns the wrong value.

abs(A)	Absolute value of A. If A is negative, returns -A otherwise returns A.
acos(A)	Arc-cosine of A. Returns the angle, measured in radians, whose cosine is A.
acosh(A)	Same as acos() but for hyperbolic cosine.
asin(A)	Arc-sine of A. Returns the angle, measured in radians, whose sine is A.
asinh(A)	Same as asin() but for hyperbolic sine.
atan(A)	Arc-tangent of (A). Returns the angle, measured in radians, whose tangent is (A).
atan2(A,B)	Arc-tangent of A/B. The two main differences to atan() is that it will return the right angle depending on the signs of A and B (atan() can only return values betwen -pi/2 and pi/2), and that the return value of pi/2 and -pi/2 are possible.
atanh(A)	Same as atan() but for hyperbolic tangent.
ceil(A)	Ceiling of A. Returns the smallest integer greater than A. Rounds up to the next higher integer.
cos(A)	Cosine of A. Returns the cosine of the angle A, where A is measured in radians.
cosh(A)	Same as cos() but for hyperbolic cosine.

cot(A)	Cotangent of A (equivalent to 1/tan(A)).
csc(A)	Cosecant of A (equivalent to 1/sin(A)).
exp(A)	Exponential of A. Returns the value of e raised to the power A where e is the base of the natural logarithm, i.e. the non-repeating value approximately equal to 2.71828182846.
floor(A)	Floor of A. Returns the largest integer less than A. Rounds down to the next lower integer.
if(A,B,C)	If int(A) differs from 0, the return value of this function is B, else C. Only the parameter which needs to be evaluated is evaluated, the other parameter is skipped; this makes it safe to use eval() in them.
int(A)	Rounds A to the closest integer. 0.5 is rounded to 1.
log(A)	Natural (base e) logarithm of A.
log10(A)	Base 10 logarithm of A.
max(A,B)	If A>B, the result is A, else B.
min(A,B)	If A<B, the result is A, else B.
sec(A)	Secant of A (equivalent to 1/cos(A)).
sin(A)	Sine of A. Returns the sine of the angle A, where A is measured in radians.
sinh(A)	Same as sin() but for hyperbolic sine.
sqrt(A)	Square root of A. Returns the value whose square is A.
tan(A)	Tangent of A. Returns the tangent of the angle A, where A is measured in radians.
tanh(A)	Same as tan() but for hyperbolic tangent.

Examples of function string:

1+2
x-1
-sin(sqrt(x^2+y^2))
*sqrt(XCoord*XCoord + YCoord*YCoord)*

VisIVO Filters

Merge Tables

This operation creates a new table from two or more existing data tables . Up to 100 tables can be merged.

Usage:

VisIVOFilters --op merge [--size HUGE/SMALLEST] [--pad value] [--out filename_out.bin] [--help] [--file] table_param.txt

Example:

VisIVOFilters --op merge --out out_table_file.bin --file tab_selection_file.txt

the tab_selection_file.txt is a file that contain a list of tables and valid columns. Wildcard "*" means all the columns of the given table.

tab1.bin Col_1
tab1.bin Col_2
tab5.bin Col_x
*tab4.bin **

This command produces a new table (*filename_out.bin* and *filename_out.bin.head*) having columns Col_1 and Col_2 from tab1.bin, Col_x from tab5.bin and all the columns of tab4.bin

Note:
--size SMALLEST default option. Produce a new table having the size of the smallest table. HUGE produces a new table having the size of the greatest table.
--pad Pads the table rows of the smallest table with the given value if HUGE size is used. Default value is 0.
--out Name of the new table. Default name is given.
--file Input table filename.

VisIVO Filters

Point Distribute

This operation creates a volume using a field distribution (CIC algorithm) on a regular mesh.

Usage:

VisIVOFilters --op pointdistribute --resolution x_res y_res z_res --points x_col y_col z_col [--field column_names] [--out filename_out.bin] [--help] [--file] inputFile.bin

Example:

VisIVOFilters --op pointdistribute --resolution 16 16 16 --points X Y Z --field Mass Temperature --out filename_out.bin --file inputFile.bin

It produces a new volume table (*filename_out.bin* and *filename_out.bin.head*) on a *16x16x16* mesh using x y and z as point coordinates.
The filter distributes the *Mass* and the *Temperature* elements of points *X Y Z*. Only the *Mass* and *Temperature* fields will be reported in the new table.

Note:
--resolution 3D mesh size.
--points Columns to be assumed for points coordinates.

--field valid columns name list to be distributed in the grid. Default value is a Constant=1 for all points.
--out Name of the new table. Default name is given.
--file Input table filename.

VisIVO Filters

Point Property

This operation assigns a property to each data point on the table. The operation performs the following:
1) It creates a temporary volume using a field distribution (CIC algorithm) on a regular mesh. The temporary file will have the filename given in --out option + "_tempPDOp.bin" or "./_tempPDOp.bin", and it is automatically cleaned by the operation itself.
2) It computes, with the same CIC algorithm, the property for each data point, considering the cells where the point is spread on the volume;
3) It saves the property in a new table or adds the field to the original input table.

This operation cannot be applied to volumes.

Usage:

VisIVOFilters --op pointproperty --resolution x_res y_res z_res --points x_col y_col z_col [--field column_name] [--append] [--out filename_out.bin] [--outcol col_name] [--help] [--file] inputFile.bin

Example:

VisIVOFilters --op pointproperty --resolution 16 16 16 --points X Y Z --field Mass --append --outcol distribute --file inputFile.bin

The filter distributes the *Mass* field of points *X Y Z and it produces a temporary volume*. Then it calculates a new field that for each X Y Z point, representing the weight value of the nearest mesh-cells where the point is distributed using the same CIC algorithm.

Note:
--resolution 3D mesh size.
--points Columns to be assumed for points coordinates.
--field valid columns name list to be distributed in the grid. Default value is a Constant=1 for all points.
--append the input table will contain the new field.
--out Name of the new table. Default name is given (ignored if –append is given).
--outcol New field column name.
--file Input table filename.

Randomizer

This operation creates a subtable as a random subsample from the input table.

Usage:

VisIVOFilters --op randomizer --perc percentage [--list parameters] [--iseed iseed] [--out filename_out.bin] [--help] [--file] inputFile.bin

Example:

VisIVOFilters --op randomizer --perc 10.0 --iseed 1 --out filename_out.bin --file inputFile.bin

produces a new table (*filename_out.bin* and *filename_out.bin.head*) of 10 percent size of the original inputFile. Values are extracted in a random sequence.

Note:
--perc Percentage (from 0.0 to 95.0) of the input file obtained in the output file. Note: only the first decimal place is considered.
--list Valid columns names of the input table. Default: all columns are included.
--iseed Specifies the seed for the random generation. Default value 0.
--out Output table filename. Default name is given.
--file Input table filename

VisIVO Filters

Select Fields

This operation creates e new table using one or more fields of a data table

Usage:

VisIVOFilters --op selcolumns --list parameters [--extract] [--delete][--out filename_out.bin] [--help] [--file] inputFile.bin

Example

VisIVOFilters --op selcolumns --list X Y --extract --out filename_out.bin --file inputFile.bin

produces a new table (*filename_out.bin* and *filename_out.bin.head*) with only *X* and *Y* columns of inputFile..

Note:
--list Valid columns names of the input table.
--extract Default option. It produces the output table including only the fields listed in the --list option.

--delete Produces the output table excluding only field listed in the --list option. This option supersedes the --extract option.
--out Output table filename. Default name is given.
--file Input table filename.

VisIVO Filters

Select Rows

This operation creates e new table setting limits on one or more fields of a data table

Usage:

VisIVOFilters --op selfield --limits filename_limits [--operator AND/OR] [--out filename_out.bin] [--help] [--file] inputFile.bin

Example:

VisIVOFilters --op selfield --limits limitsfile.txt --operator AND --out filename_out.bin --file inputFile.bin

The limitsfile.txt file must have the following structure. A valid column name and an interval indicating the extraction limits:

X 20.0 30.0
Y 10.0 20.0
Z 0.0 10.0

The command produces a new table (*filename_out.bin* and *filename_out.bin.head*) that contains all the data points of the inputFile table (all columns will be reported) where *X*=[20.0,30.0] AND *Y*=[10.0,20.0] AND *Z*=[0.0,10.0]

Note:
--limits A file that has three columns: a valid column name and an interval indicating the extraction limits.
--operator Limits on all fields listed in --limits option file are combined by default with logic AND operator. If this option is given with OR value the field limits are combined with logic OR operator
--out Output table filename. Default name is given.
--file Input table filename.

VisIVO Filters

Show Table

This operation produces an ASCII table of the selected field of the first number of rows (or of all rows)

Usage:

VisIVOFilters --op showtable [--field column_name] [--numrows num_of_rows] [--rangerows fromRow toRow] [--width format_width] [--precision format_precision] [--out filename_out.txt] [--help] [--file] inputFile.bin

Example:

VisIVOFilters --op showtable --field X Y Z --numrows 100 --out filename_out.txt --file inputFile.bin

produces an ascii file (filename_out.txt) that contains the first 100 data points of the *X,Y* and *Z* fields of the *inputFile* table.

Note:
--field Valid columns names. Default value of ALL columns will be reported.
--numrows Number of Rows in the Ascii output file. Default value is equal to the number of Rows of the input table.
--rangerows Rows range of the inputFile that will be reported in the ascii output file. Default range is equal to all the rows of the input table. It is ignored if numrows is specified.
--width Field width in the Ascii output file. Default value is given.
--precision Field precision in the Ascii output file. Default value is given.
--out Output ascii filename. Default name is given.
--file Input table filename.

VisIVO Filters

Statistic

This operation produces average, min and max value of field and creates an histogram of fields in the input table

Usage:

VisIVOFilters --op statistic --list columns_name [--histogram [bin]] [--range min max] [--out result.txt] [--help] [--file] inputFile.bin

Example:

VisIVOFilters --op statistic --list X Y --histogram 1000 --range 10.0 100.0 --out result.txt --file inputFile.bin

produces *min,max* and *average* values printed in stdout. The command also produces a *result.txt* file that gives the histogram values of X and Y in the range 10.0,100.0 with 1000 bins.

Note:
--list A valid list of columns name
--histogram produces an histogram ascii file with a given number of bins. If the bins number is not specified, the default value is fixed to 10% of the total rows of the input table.
--range produces the results only inside the specified interval.
--out Output ascii filename with histogram
--file Input table filename.

An error is given if there are no data in the specified range.

VisIVO Filters

Visual

This operation creates an eventually randomized new table from one or more input tables. All the input tables must have the same number of rows. The new table can be used with VisIVOViewer. The operation cannot be applied to volume tables.

Usage:

VisIVOFilters --op visualop [--size number_of_elements] [--out filename_out.bin] [--help] [--file] tab_selection_file.txt

Example:

VisIVOFilters --op visualop --out filename_out.bin --file tab_selection_file.txt

The tab_selection_file.txt is a text file that contain a list of tables and valid columns. Wildcard "*" means all columns of the given table.

tab1.bin *Col_1*
tab1.bin *Col_2*
tab5.bin *Col_x*
tab4.bin *

The command produces a new table (*filename_out.bin* and *filename_out.bin.head*) having columns *Col_1* and *Col_2* from *tab1.bin*, *Col_x* from *tab5.bin* and all the columns of *tab4.bin*. If the row number of the input tables exceeds *8000000* elements, the output file will be limited to *8000000* randomized sampled rows

Note:
--size Number of max rows in the output table. Default is the minimum between 8000000 rows and the rows of input tables. The input tables must have the same number of rows.
--out Output table filename. Default name is given.
--file Input text file with a list of tables and columns

VisIVOServer 1.0

(subversion 1.0.0.4)

VisIVO Viewer

VisIVOViewer creates views from the input data file. The input data file must be in the Internal Binary format. The input data file must fit the available RAM. VisIVOViewer produces five png images. The first four default images are generated with the following fixed values:

Azimut	0	90	0	45
Elevation	0	0	90	45
Zoom	1			

The last image is given with values fixed by the user

GENERAL SYNTAX

VisIVOViewer --help

produces a general help

VisIVOViewer <options> inputFile

produces images from the input file.

The alternative command is:

VisIVOViewer parameterFile

GENERAL OPTIONAL OPTIONS

--out [filename] Default output filenames are VisIVOServerImage0.png, VisIVOServerImage1.png .... Note: the output name is always completed with *0.png 1.png 2.png and 3.png* if default images are produced. Anyway, the output file name (image) is completed with *.png* extension

--nodefault Default images are not created.

--cycle [filename] (optional) VisIVO viewer will produce a sequence of images reading data of azimuth elevation and zooming from the file given with this parameter. This option will

prevent the production of default images while the camera position and zoom factor of the line command will be ignored. The file for the cycle must contain three ascii values for each row. Blank line or rows with less than three values will be ignored. VisIVO will produce one image for each valid row in the file. This file can be created with the tool *VisIVOUtils --op createpath* option. The output filenames will have the same root-name as those given in *--out* parameter plus a progressive number starting from 0 (or the value given in the --cycleoffset option), with png file extension. A new file listing all images is created. The file has the fixed name VSCycleImages#.txt in the output director, where # is the value of --cycleoffset option.
--cycleoffset [value] (optional) is the value for the progressive number of files produced with the cycle option. The default value is 0.

--cycle_skip_from [value] (optional) skips the first number of lines in the cycle file
--cycle_skip_to [value] (optional) reads up to the given line in the cycle file
(E.g.: --cycle_skip_from 30 --cycle_skip_to 34, read lines 31,32,33 and 34 (included))

--camazim [double] (optional) Fixes the camera azimuth position
--camelev [double] (optional) Fixes the camera elevation position.
The allowed range for the camera elevation is [-90 , 90].
If the given camera elevation value is out of the valid range, the elevation is set at the boundary. Ex: --camelev 100 will automatically be changed with --camelev 90.
--zoom [double] (optional) Zooming factor. A value greater than 1 is a zoom-in, a value less than 1 is a zoom-out *(--zoom 2.0)*

--color (optional) Uses the LUT
--colortable [name] (optional) Selects the LUT *(--colortable default) or (--colortable paletteFilename)*

The following prefixed LUT name can be given

default default_step efield glow gray min_max physics_contour pure_red pure_green pure_blue run1 run2 sar temperature tensteps volren_glow volren_green volren_rg volren_twolevel

NOTE:
If the **--colortable** option does not contain a prefixed value, VisIVOViewer assumes that an external filename is given as for this parameter. The file must exist in the current directory or the path must be specified. Lut is an ASCII file: each line contains the RGBA values or only RGB values. Lines with lower than 3 values are ignored.
Valid line files are:

255 170 170

or

255 170 170 125

The following kinds of visualizations are available: data points, isosurfaces, volumes and vectors.

The alternative command allows VisIVOViewer to read all options (including the Splotch options) from a parameter file. Lines starting with # are comments. Splotch parameter are included in this file.
An example of this file is the following:

```
############## VisIVO SECTION
######## Sect 1 General
volume=no              ← no|yes default value: no
vector=no              ← no|yes default value: no
splotch=yes            ← No parameter splotch file must be given: only no (default) or yes
input=u2.bin
out=outFilename
#cycle=cicleFilename
#cycleoffset=0
#cycle_skip_from0
#cycle_skip_to=0
######## Sect 2 Points and vectors
x=X
y=Y
z=Z
#vx=VX
#vy=VY
#vz=VZ
#scale=yes
#######
########### Sect 3 Volume
#vrendering=yes
#isosurface=no
#slice=no
#shadow=no
#vrenderingfield=ColumnName
#slicefield=ColumnName
#sliceplane=x
#sliceposition=0.0
#isosurfacefield=ColumnName
#isosurfacevalue=120
################ Sect 5 Camera
camazim=20
camelev=20
zoom=1.5
nodefault=yes                ← put yes if you do not want the default images
#largeimage=no
################ Sect 4 Colour
color=yes
colorscalar=X
colortable=default
opacity=0.666
#logscale=no
###########  Sect 5 Glyphs
#glyphs=sphere
```

```
#scaleglyphs=no
#radius=1.0
#radiusscalar=ColumnName
#height=1.0
#heightscalar=ColumnName

##############  SPLOTCH SECTION
# Visualization properties for gas
gas=Y
label_intensity=XXXX
label_color=lgdensity
label_hsml=HSML2
log_intensity=FALSE
log_color=FALSE
#
min_int=0
max_int=2
#
min_col=20.0
max_col=60.0
#
hsml_sl= 0.2
hsml_fac=1
brightness_gas=     0.1
gray_absorption_gas= 1.0
#
stars=N
#
resolution=800
ycut0=0
ycut1=800
boostcolors=FALSE
colourbar=TRUE
#sky_x=0
#sky_y=1
#sky_z=0
```

VisIVO Viewer

Data Points

VisIVOViewer creates data points views from the input data file. The Input data file must be in Internal Binary format. The input data file must fit the available RAM.

SPECIFIC DATA POINTS OPTIONS

--x [field] (optional) Selects the first coordinate *(--x x-coordinate)*
--y [field] (optional) Selects the second Coordinate *(--y y-coordinate)*
--z [field] (optional) Selects the third Coordinate *(--z z-coordinate)*
--scale (optional) Enables data normalization. It always allows you to visualize a cubic region even if the coordinates system has different scales.

The field names containing *X,Y,Z* or *RA,DE* and *Mag* are assumed to be default values for the x y z system, or the first three table columns, if these options are not given. A warning message will be given. NOTE: it is strongly recommended to fix these parameters to prevent unpredictable behavior.

--colorscalar [field] (optional) Selects the field for the lut *(--colorscalar scalar0)*
--logscale (optional) Uses the logarithmic scale for the LUT. If the select field has values <=0 this option is ignored and the linear scale will be used.

--glyphs [name] (optional) Data points are displayed with different geometrical form. The following forms are available: *pixel sphere cone cylinder cube (--glyphs sphere)*. This option has no effect if the data point number is more than 1000.

--radius (optional) Radius of the geometrical form *(--radius 2.0)*
--height (optional) Height of the geometrical form (where applicable) *(--height 5)*
--opacity [double] (optional) Data points opacity. Default value 0.66 *(--opacity 0.03)*

--scaleglyphs (optional) Enables the geometrical form to be scaled with a scalar field.
--radiusscalar [field] (optional) Sets the scalar field for radius scaling *(--radiusscalar scalar0)*
--heightscalar [field] (optional) Sets the scalar field for height scaling *(--heightscalar scalar1)*

Examples

LUT usage

VisIVOViewer --x X --y Y --z Z --color --colorscalar scalar0 --colortable temperature --logscale /home/user/inputFile.bin

Normal GLIPHS

VisIVOViewer --x X --y Y --z Z --glyphs cone --radius 1 --height 2 /home/user/inputFile.bin

Scaled GLYPHS

VisIVOViewer --x X --y Y --z Z --glyphs cone --scaleglyphs --radiusscalar scalar0 --heigthscalar scalar1 /home/user/inputFile.bin

VisIVO Viewer

Volumes

VisIVOViewer creates a volume view of data points from the input data file that contains a volume. The input data file must be in Internal Binary format and must have the number of mesh elements on each dimension. The input data file must fit the available RAM. A volume can be visualized with the volume rendering technique, with an isosurface or with slices.

SPECIFIC VOLUME OPTIONS

--volume (optional) enables volume visualization

VOLUME RENDERING Visualization OPTIONS

--vrendering (optional) enables volume rendering view. The volume rendering view is the default when --volume is given.
--vrenderingfield [field] sets the scalar to be represented in the view. *(--vrenderingfield scalar0)*
--shadow (optional) enables shadow view in the rendering view.

Example

VisIVOViewer --volume --vrendering --vrenderingfield density –colortable temperature /home/user/inputFile.bin

ISOSURFACE Visualization OPTIONS

--isosurface (optional) enables isosurface view
--isosurfacefield [field] (optional) sets the scalar to be represented in the view *(--isosurfacefield scalar0)*
--isosurfacevalue [field] (optional) fixes the isocontur value: from 0 to 255 *(--isosurfacevalue 200)*

Example

VisIVOViewer --volume --isosurface --isosurfacefield density --isosurfacevalue 200 /home/user/inputFile.bin

SLICER Visualization OPTIONS

--slice (optional) enables slice view
--slicefield [field] (optional) sets the scalar to be represented in the slice view *(--slicefield scalar0)*
--sliceplane [plane] (optional) sets the plane to be represented in the view *(--sliceplane y)*
--sliceposition [position] (optional) sets the plane coordinate position to be represented in the view *(--sliceposition 5.0)*

NOT YET DEVELOPED

Example

VisIVOViewer --volume --–slice --slicefield density--sliceplane x –sliceposition 3.0 –color --colortable default /home/user/inputFile.bin

VisIVO Viewer

Vectors

VisIVOViewer creates a view of vectors created from the input data file that contains data points. The input data file must fit the available RAM.

TO BE DEVELOPED

SPECIFIC VECTOR OPTIONS

--vector (optional) enables vector visualization
--vx [field] (optional) First component of the vector. Any field in the input file table(--vx X)
--vy [field] (optional) Second component of the vector. Any field in the input file table (--vy Y)
--vz [field] (optional) Third component of the vector. Any field in the input file table (--vz z)

Example

NOT YET DEVELOPED

VisIVO Viewer

Splotch

VisIVOViewer integrates the main basic Splotch functionalities (citation). VisIVO uses, up to now, only the gas visualization of Splotch. Stars will be integrated in the future. Some default values are changed. VisIVO Splotch cannot be used for Volume visualization. This option can be used only to visualize Data Points and all the options of that section can be used (e.g. --x, --y, --z, --camazim, –camelev, --zoom, --out, --cycle .. etc.)

SPECIFIC SPLOTCH OPTIONS

--splotch *splparafile*

The splparafile is the same file of the Splotch program but some news are included and some lines are ignored.

Example

VisIVOViewer --splotch /home/user/splotchfile.par --x X --y Y --z Z --color --colortable mypalette.pal --cycle /home/user/mypath.file /home/user/inputFile.bin

The splparafile parameter file has the following sections. All the original fields are reported in the following even if ignored by VisIVO. Comment line starts with # character. The user must give only the lines that he wants to specify. FALSE and TRUE values must be given in capital case.
The Splotch option can be used with --cycle option.

a) Input Output Section

```
# Input/Output options
#infile=inputFilename.bin
#outfile=outputFilename.bin
#numfiles=1
swap_endian=FALSE
```

VisIVO ignores the fields infile and outfile numfiles. The infile must be given an input file of VisIVOViewer and outfile with the --out option. The image with tga extension will be produced. No extension tga must be given in the filename.

b) Visualization properties for gas

```
# Gas properties
gas=Y
```

c) Gas Intensity Field

```
# Intensity field
```

label_intensity=XXXX
log_intensity=FALSE
min_int=0
max_int=2

The label_intensity field is a column of the inputfile table. The columns values are used to give an intensity to each data point in the visualization.
If the column is not found (or the field is not present) a default value of 1.0 is given to each data point. The log_intensity field is a boolean (TRUE or FALSE) to use a log scale for the intensity. The default value is false. This value is set to *false* if *label_intensity* is not a valid field and the default value is used.
The min_int and max_int values represent the maximum and minimum value (offset) for the intensity field to be used in the visualization.
Default values are set as follows:
- if the default value of 1.0 is used for the intensity, min_int and max_int have 0 and 2 values, respectively.
- if a valid column is used the min_int and max_int have the minimum and maximum value in the column, respectively

d) Gas Color Field
# Color Field
label_color=density
log_color=FALSE
min_col=20.0
max_col=60.0

The label_color field is a column of the inputfile table. The columns values are used to give a color to each data point in the visualization.
If the column is not found (or the field is not present) a default value of 1.0 is given to each data point. The log_intensity field is a boolean (TRUE or FALSE) to use a log scale for the intensity. The default value is false. This value is set to *false* if *label_color* is not a valid field and the default value is used.
The min_col and max_col values represent the maximum and minimum value (offset) for the color field to be used in the visualization (minimum and maximum value in the LUT).
Default values are set as follows:
- if the default value of 1.0 is used for the color, min_color and max_color have 0 and 2 values, respectively.
- if a valid column is used the min_color and max_color have the minimum and maximum value in the column, respectively

e) Smooting Length field
# HSML Field
label_hsml=HSML
hsml_fac=1
hsml_sl= 0.2

The label_hsml field is a column of the inputfile table. The columns values are used as the smoothing length for the visualization.

This paramater has a strong influence on data visualization and on computing time. No adequate values could also generate an image that does not appear to be correctly focused. If the column is not found (or the field is not present) a default value of hsml_sl is given to each data point. The hsml_sl default value is set as the ratio between the average edge of the data box and 8 * cubic root of the number of points. The latter value was derived from some statistical considerations.
The hsml_fac is a correction factor to be applied to the values contained in the label_hsml field. The hsml_fac default value is 1.0.

NOTE: If the smoothing length value is too low (depending on the box coordinates) a flat image can be produced. If this value is too high (depending on the box coordinates) the computing time will increase and flat images can be produced.
However it is recommended to try different values for hsml to produce the desired images.

f) Color properties
filegas_palette=mypalette.pal
gray_absorption_gas= 0.2
brightness_gas= 0.1

The filegas_palette can be used to read an external LUT. **This option is obsolete and it is strongly recommended to set the LUT table with the --color and --colortable options**.
The filegas_palette however refers to an external file that contains a valid Splotch palette. This palette is used if this row is present in the parameter file.
The file must have the following lines:

Maypalette
0100
5
0 0 255
255 170 170
128 255 128
0 170 85
255 0 0

The first line is the palette name, the second line is the palette version . These two values have no effect on the effective image color.
The third line contains the number of lines that will be considered for the palette. The following lines contain the RGB values.
The gray_absorption_gas field represents the gray absorption of the points in the image. The default value is 0.2.
The brightness_gas field represents the luminosity of the points in the image. It has a similar effect of the opacity option. The default value is 0.1

IMPORTANT NOTE: The brightness_gas field value can change significantly the produced image. A high value can give a white image, a low value can produce a black image. The optimal value strongly depends on the dataset. Nevertheless it is recommended to try different values for hsml to produce the desired image.

g) Windows Properties
resolution=800
ycut0=100
ycut1=700
boostcolors=FALSE
colourbar=TRUE

The resolution factor is the resolution of a square image: the default value is 800 that will produce an 800x800 image. The boostcolors is to Boosting colors. The default value is false. The ycut0 parameter will cut the image from the bottom up to ycut0. The ycut1 parameter will cut the image from ycut1 to the top of the image. The colourbar is used to print the colorbar in the image. The default value is TRUE.

h) Geometrical setup

NOTE: The GEOMETRY_FILE Splotch option must not be enabled for the VisIVOViewer compilation.

i) Camera position
Azimuth zoom and elevation are fixed from --camazim --camelev and --zoom options
Clipping planes are determined using the VTK Camera.
NOTE: no default images are produced with the Splotch method.

Rolling factor is still not enabled in VisIVO. If you need to set it, the Splotch parameter file can include the position for the rolling factor as follows:

sky_x=0
sky_y=1
sky_z=0

VisIVOServer 1.0

(subversion 1.0.0.4)

VisIVO Utilities

VisIVO Utilities is a tool that creates intermediate data that will be used by the other components of VisIVO Server. These data could consist of a sequence of values or files extracted from VisIVO Binary Tables.

GENERAL SYNTAX

VisIVOUtils --help
produces a general help

VisIVOUtils --op utilitycode --help
produces the help of a specific utility

VisIVOUtils --op utilitycode <options>
runs the utility

The following utilities are available

Create path
creates an ascii file containing three values for each row with azimuth elevation and zoom

VisIVO Utilities

Create path

This utility creates an ascii files containing three values for each row with azimuth elevation and zoom. The file can be used by VisIVOViewer --cycle to produce a sequence of png images to be mounted in a movie.

Usage:

VisIVOUtils --op createpath [--azimuth from to] [--elevation value] [--zoom value] [--zoomend [stepframe]] [--framesec value] [--length value] [--out filename] [--help]

Example:

VisIVOUtils --op createpath --azimuth 0.0 60. --elevation 0.0 10.0 --zoom 1.0 1.5 --zoomend --length 20 --out my_cycle.par

The utility produce 10 values (default value) for each second. The file my_cycle.par with 204 values as follow:

```
0.0   0.0   1.0
0.3   0.05  1.0
0.6   0.1   1.0
......
......
60.0  10.0  1.0
60.0  10.0  1.2
60.0  10.0  1.4
60.0  10.0  1.6
```

Note:

--azimuth Movement *from to*. Default values 0.0 and 0.0.
--elevation Movement from to. Default values 0.0 and 0.0. Valid range [-90,90]. Values outside this interval are automatically set to the near extreme: Ex.: *--elevation -85 100* will be modified with *–-elevation -85 90*.
--zoom Zoom from to. Default values 1.0 and 1.0. A zooming factor <1 represents a zoom in a zooming factor >1 represent a zoom out. Negative value are ignored
--zoomend The zoom is given at the end. The value step-frame represent the step for zooming . Default step-frame is 0.2 If this option is given priority with zoom will be ignored. The final zooming is added to global the length.
--framesec Number of frame values for each second. Default value is 10 sec.
--length Value in seconds. Default value is 10 sec.
--out Output filename. Default filename cycle.par. The file is opened in append mode.